# A Forager Adjustment Strategy Used by the Bees Algorithm for Solving Optimization Problems in Cloud Manufacturing

Xie, Yongquan; Zhou, Zude; Liu, Quan; Xu, Wenjun; Ji, Chunqian; Liu, Ping; Tian, Sisi; Pham, Duc

*DOI:*
10.1115/MSEC2015-9255

# A FORAGER ADJUSTMENT STRATEGY USED BY THE BEES ALGORITHM FOR SOLVING OPTIMIZATION PROBLEMS IN CLOUD MANUFACTURING

**Yongquan Xie[1,2], Zude Zhou[1,2], Duc Truong Pham[3], Quan Liu[1,2], Wenjun Xu[1,2], Chunqian Ji[3], Ping Lou[1,2], Sisi Tian[1,2]**

[1]School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China

[2]Key Lab. of Fiber Optic Sensing Technology and Information Processing, Ministry of Education, Wuhan University of Technology, Wuhan 430070, China

[3]School of Mechanical Engineering, University of Birmingham, Edgbaston, Birmingham, B152TT, U.K.

xyqwhut0728@126.com, zudezhou@whut.edu.cn, d.t.pham@bham.ac.uk, quanliu@whut.edu.cn, xuwenjun@whut.edu.cn, c.ji@bham.ac.uk, louping@whut.edu.cn, tiansisi@whut.edu.cn

## KEYWORDS

Swarm intelligence, Bees Algorithm, forager adjustment strategy, intelligent optimization, cloud manufacturing

## ABSTRACT

Intelligent technologies have become increasingly important in manufacturing nowadays. Optimal service management and allocation in current cloud manufacturing model are impossible without applications of appropriate intelligent tools. The Bees Algorithm (BA) is a swarm-based intelligent optimizer that provides support for smart decision-making process in manufacturing models. A novel forager adjustment strategy (FAS) is proposed in this paper to manage the forager division in the algorithm, so as to make the entire colony perform with higher efficiency. The proposed FAS based Bees Algorithm (FAS-BA) is able to realize flexible allocation of its forager resources between different roles in accordance with the solution fitness sampled by current scout population. The proposed algorithm is presented in detail. Experiments are conducted based on a set of well-known benchmark functions and a case study. Comparisons between FAS-BA and an improved Bees Algorithm are made to highlight the effectiveness of FAS. The results demonstrate that the proposed algorithm requires less function evaluation cost than the improved version but is capable of obtaining at least the same optimal solution to a problem.

## INTRODUCTION

Cloud manufacturing is a new generation service-oriented networked manufacturing model that provides users with distributed manufacturing resources and capacities. In this context, the notion of resource service optimal allocation describes the linking of a physical order or product to resource services and capacities, while combining information and rules governing the way the order is likely to be fulfilled. Due to the availability of internet and service virtualized techniques, accessible services in cloud service pool proliferate. In addition, internal and external changes in a cloud become frequent and unpredictable, and the number of service requests and users is becoming increasingly large, making the optimal service allocation and flexible supply chain management difficult problems. An intelligent process in cloud manufacturing enables a system to optimize the management of supply and production chain, the allocation of physical and virtualized resource services, the expense of labor and raw materials, etc. Artificial intelligence is a core enabling technology for cloud manufacturing to solve the emerged problems. Swarm-based intelligent optimization algorithm is a powerful probabilistic heuristic procedure for the search of optimal solution based on swarm intelligence, which defines much of the natural world from a flock of birds to a colony of ants. It is featured by a distributed intelligence and self-healing infrastructure that will have a major impact on a highly mobile cloud environment of distributed manufacturing resources. Swarm-based optimization algorithms (SOAs) find optimal or near-optimal solutions to complex problems such as resource service composition, job scheduling, and assembly planning. The optimizing behavior of individuals in the algorithm translates particularly well into technological applications in cloud computing or manufacturing. Members of SOAs include Ant Colony Optimization (ACO) [1], Genetic Algorithm (GA) [2], Particle Swarm Optimization (PSO) [3], Biogeography-based Optimization (BBO) [4], Artificial Bee Colony (ABC) [5], etc. Over the last two decades, these algorithms have been used to transform different aspect of technology, igniting new developments in areas from robotics in its first applications to cloud manufacturing technology today. They have been shown good quality in dealing with optimization problems especially those have shown to be NP-hard and cannot be solved in bounded polynomial computation time.

The Bees Algorithm was originally proposed by Pham et al [6-7]. It is characterized by a combination of neighborhood search and global search. To this day it has gone through several variations. The Bees Algorithm optimize an objective

problem in a single run through parallel computing, which usually incurs a large number of solution samplings and fitness evaluations. Therefore efforts have consistently been made to enhance its search efficiency. The neighborhood shrinking and site abandonment strategies were introduced for faster search speed and higher accuracy [8]. The pheromone-based recruitment was employed also to speed up the search [9]. The fuzzy greedy selection of local search sites was put forward to reduce the algorithm's parameters while improving its robustness and self-organizing ability [10].

The forager adjustment strategy proposed in this paper is intended to promote the optimal solution search efficiency of the algorithm. This strategy regulates the labor division in a colony. Like the Bees Algorithm, this strategy also takes inspiration from a biological phenomenon. In a honeybee colony, the proportion of bees designated for different food foraging tasks adapts to the changing habitat. Each forager in the colony specializes in one task at a time. This is a ubiquitous feature in the bee's world. The colony's external supply of food can vary greatly from time to time with the flower clusters flourishing and withering. To successfully survive in the diverse and constantly changing natural environments, the colony allows the transfer of foragers from one role to another [11]. The colony pulls up its standard of food quality when there are plenty of food sources available, and then only the sources bearing superior nectar or pollen are favored. More foragers (called recruiters or recruited followers in the following text) are devoted to explore those high-quality sources. In the colony, the number of bees for discovering new sources (called scouts or scout bees in the following text) consequently declines. However, when it becomes very difficult in finding a source that provides favorable food, in later autumn for example, the colony lower its food selecting standard. Meanwhile, the bees that used to be recruiters are dispatched to play the role of scouts in order to find more available sources, resulting to the decrease in the number of recruited followers. In this way the colony could adapt itself on the one hand to its own time-dependent demands for nectar and water from the habitat, and on the other hand to the dynamic natural conditions like temperature and the amount of food obtainable.

The paper is organized as follows: Section 2 introduces the bee's behaviors in nature and reviews the basic Bees Algorithm. Section 3 details the proposed strategy. The presentation of the controlling parameters in new schema and the procedures of FAS-BA are also included in this section. Section 4 contrasts the proposed algorithm versus an improved version. Analyses and discussions about the experiment are presented. A case study that demonstrates how the algorithm can be effective in resource service composition in cloud manufacturing environment is involved in Section 5. Finally, Section 6 concludes the paper and gives suggestions for future work.

## THE BEES ALGORITHM

This section briefs the bee's foraging behaviors in nature and the basic version of the Bees Algorithm. Without loss of generality, it will be assumed in the rest of this paper that the optimization problem requires the minimization of the objective function.

### Bees Foraging Behaviors in Nature

A honeybee colony is able to perform foraging action over an immense area around the nest of the colony during the harvest season. It extends part of its population over long distance to scout the fields [12]. Scout bees move randomly looking for food sources. Once they discover potential sites where nectar or pollen is abundant and in high quality, they return to the hive and perform a "waggle dance" to advertise the sites they discovered. Biologists believe the purposed of this mysterious dance is to broadcast the information including the direction, distance and quality of the sites. After observing the dance, the bees waiting in the hive follow the scout bees to the sites and begin to collect the nectar intensively, and this part of population is known as the recruited followers. The number of followers depends on the quality rating of the food source. The sites that provide richer and more easily available nectar attract a larger number of followers than other sites. While some scout bees are broadcasting the information of food sources and recruiting followers, those scout bees fail to find valuable sites continue to fly randomly over the region as wide as they can in the hope of discovering potential sources. This intensive-and-extensive food discovering and collecting mechanism allows the honeybee colony to optimize the efficiency of food gathering. An intelligent optimization process in engineering problems today is analogous to this mechanism as the honeybees strive to collect the most favorable food for sustainable existence.

### The Basic Bees Algorithm

Every possible solution in the searching space is thought of as a food source and represented by a site in the algorithm. The Bees Algorithm starts by sending a number of scout bees to randomly sample the space. The fitness of the sampled sites denotes the quality of food and is evaluated by the objective function. The sampled sites are ranked into a queue in conformity to their fitness value. Among the scout bees, only those landing at the top ranked sites have the qualification for recruiting followers. The recruited followers search the fitness landscape in the neighborhood of the top ranked sites. This process is called the local search (or neighborhood search). The scout bees arriving at the sites that are not qualified for neighborhood search do not have followers. They continue to perform random sampling in the entire solution space in the next iteration. This process is called the global search. The Bees Algorithm locates the promising sites and selectively exploits their neighborhoods while still randomly explores the entire solution space. By doing so it applies to the search for global optimal solution to the objective function.

The algorithm requires a number of parameters to be set, namely: number of scout bees ($n$), number of best sites selected from the n visited sites for local search ($m$), number of elite sites out of the m selected site for a more intensive local search ($e<m$), number of recruited followers for local search around

elite sites(*nre*), number of bees recruited for local search around the rest m-e best sites (*nrb*<*nre*), initial size of neighborhood (*ngh*), stagnant cycles for site abandonment (*stlim*) and the stopping criterion for the algorithm to terminate. The literature [6] details the pseudo code of the basic Bees Algorithm, and more recent researches on the Bees Algorithm are provided in [13-16].

## THE PROPOSED FAS-BA

### New Parameter Schema

The FAS is introduced to adaptively allocate forager resources in the honeybee colony for various tasks. In FAS-BA, foragers are also identified as scouts and recruited followers. However, instead of specifying their quantities directly in the initialization step, they are set as percentages representing their respective proportions in colony. This will be more convenient to realize forager reallocation in the algorithm. Table 1 explains the parameters in new schema. As can be seen, the colony size is preset as a decimal integer while the number of scout bees and recruited followers in each generation need to be determined by their individual proportions in the colony. The *ngh* and *stlim* remain the same as they are in the basic Bees Algorithm.

TABLE. 1 PARAMETERS OF FAS-BA

| Name | Description |
|---|---|
| *nc* | colony size |
| *sp* | proportion of scouts in the colony |
| *bp* | proportion of best sites in all sampled sites |
| *ep* | proportion of elite sites in best sites |
| *erb* | ratio of recruiters for exploring elite site against recruiters for the best sites |
| *ngh* | initial neighborhood size |
| *stlim* | limit of stagnation cycles for site abandonment |

### The Mechanism of FAS

The basic principle of FAS is: reduce the scout bees by transferring some of them to play as recruited followers for a more intensive local search if the current scout generation discovers very profitable site, whereas increase the scout bees by designating some followers to perform as scouts if current generation fails to identify the sites bearing noticeable fitness values. This puts forward a need to ascertain whether there exists some sites whose finesses stand out the other sites sampled by current scout generation. Two FAS procedures named FAS1 and FAS2 are developed to make the judgment and realize the decision-making loop. Their respective pseudo codes are demonstrated in Algorithm 1 and 2.

Algorithm 1: FAS1

| | |
|---|---|
| Input: | sampled sites |
| Step 1: | sort the sites in fitness descending order; |
| Step 2: | IF mean(*fitness*(1),*fitness*(2))>1.5×*fitness*(λ·nc·sp), reduce the scout proportion; |
| Step 3: | ELSE IF mean(*fitness*(1), *fitness*(2))<1.5×*fitness*((1-λ)·*nc*·*sp*), increase the scout proportion; |
| Step 4: | ELSE scout bees remain unchanged; |
| Step 5: | END IF |
| Step 6: | calculate the number of foragers and allocate them to respective tasks. |
| Output: | updated the number of scouts and recruited followers |

Algorithm 2: FAS2

| | |
|---|---|
| Input: | sampled sites |
| Step 1: | sort the sites in fitness descending order; |
| Step 2: | IF mean(fitness of all sites )>*fitness*(λ·*nc*·*sp*), reduce the scout proportion; |
| Step 3: | ELSE IF mean(fitness of all sites)<*fitness*((1-λ)·*nc*·*sp*), increase the scout proportion; |
| Step 4: | ELSE scout bees remain unchanged; |
| Step 5: | END IF |
| Step 6: | calculate the number of foragers and allocate them to respective tasks. |
| Output: | updated the number of scouts and recruited followers |

In the pseudo codes of FAS1 and FAS2, the function *fitness*(*i*) returns the fitness value of the *i*th site. The fitness of a site is calculated as $fitness(i) = -f(i)$, meaning it equals to the negative objective function value. A fitness factor λ is put forward. It controls the sensitivity of the forager regulation to the current fitness values of the sampled sites. In Step 1, all the sites visited by the current scout generation are ranked in a descending sequence according to their fitness. The first site in the sequence has the highest fitness but the smallest sampled function value and is therefore more favorable. Step 2 is an important step that decides whether the sites with high fitness in the sequence are distinctive enough to attract more recruited followers. In Step 2 of FAS1, the average fitness of the top two sites is calculated and compared with the one chosen by the fitness factor λ. Normally λ falls into the interval of $(0, {}^1/_2)$, hence the site being selected locates in the first half of the ordered site sequence (because the value of λ·*nc*·*sp* is definitely smaller than half of the scout number when $\lambda<{}^1/_2$). If the average fitness is larger than the chosen fitness after scaling, it is confident to believe that the sites with outstanding fitness have been sampled and are legitimate for attracting more recruited followers. In Step 2 of FAS2, the average fitness of all sites is taken into account and compared with the site chosen by λ. If the average fitness is greater than the fitness of the chosen site, the distinctive sites is said to exist in the current site sequence and thus deserve more intensive exploration by designating some scout bees as recruited followers. Step 3 is of equivalent importance to Step 2 for regulating the foragers. It determines whether the current site sequence does not contain significant fitness. If not, a more extensive search in the entire solution space rather than focusing on a few sampled sites would be fruitful. The forager regulation in Step 3 adopts a reversed logic pattern to Step 2. The forager division remains unchanged if neither condition in Step 2 or Step 3 is satisfied. This step protects the forager division from being overly

fluctuant. This is embodied in Step 4. In Step 6, the numbers of bees playing different roles are calculated from their updated proportions. Basically, the FAS1 and FAS2 rely on the dispersion of the fitness values sampled by the current scout population to fulfill the forager reallocation.

Based on the proposed strategy, the pseudo code FAS-BA is given in Algorithm 3. The introduced FAS is implemented after the algorithm finishes the global search and before the start of a new iteration cycle.

Algorithm 3: FAS-BA

| | |
|---|---|
| Input: | objective function |
| Step 1: | initialize the scout bees by sampling sites randomly in the solution domain |
| Step 2: | waggle dance (recruit followers) |
| Step 3: | local search, implement neighborhood shrinking and site abandonment strategy |
| Step 4: | global random search |
| Step 5: | implement FAS |
| Step 6: | if stopping criteria are not met, go to Step 2, otherwise terminate the algorithm |
| Output: | optimal solution |

## EXPERIMENT AND DISCUSSION

### Experiment Setup

A set of 16 well-known benchmark functions is employed to evaluate the performance of the proposed algorithm. These functions are listed in Table 2, where *D* represents the function dimensionality, the search range specifies the lower and upper boundaries of variables that compose a solution, and the optimum column gives the actual optimal solutions that have already been identified by other mathematic methods. The proposed FAS-BA is compared with the Bees Algorithm enhanced by neighborhood shrinking and site abandonment strategies [8], which is called EBA in the rest of this paper. All the algorithms are run 100 times for each function. The optimal solution is said to be found successfully when the difference between the function values of the best-fit solution found by the algorithm and the real optimum is less than 0.0001. Function f6 has more than one global minimum and locating any one of them is considered a successful trial. For each function, the algorithms run with the same colony size to make sure they require equivalent function evaluations in every iteration cycle. Therefore, the algorithm demanding the least iteration cycles to locate the optimal solution outperforms the others. The initial parameter settings are given in Table 3. The fitness factor $\lambda$ for forager adjustment is set to $^1/_3$, $^1/_4$, $^1/_5$ and $^1/_6$ to test the performance respectively.

TABLE. 2 BENCHMARK FUNCTIONS USED IN THE EXPERIMENT

| No. | Name | *D* | Search range | Optimum |
|---|---|---|---|---|
| f1 | Easom | 2 | [-100, 100] | f(x*)=-1, x*=($\pi$, $\pi$) |
| f2 | Goldstein&Price | 2 | [-2, 2] | f(x*)=3, x*=(0, -1) |
| f3 | Griewank | 2 | [-512, 512] | f(x*)=0, x*=(0, 0) |
| f4 | Schwefel | 2 | [-500, 500] | f(x*)=0, x*=(420.9687, 420.9687) |
| f5 | Martin&Gaddy | 2 | [0, 10] | f(x*)=0, x*=(5, 5) |
| f6 | Branin | 2 | [-5, 15] | f(x*)=0.397887, x*=(- $\pi$, 12.275), x*=( $\pi$, 2.275), x*=(3 $\pi$, 2.475) |
| f7 | Rastrigin | 2 | [-5.12, 5.12] | f(x*)=0, x*=(0, 0) |
| f8 | Ackley | 2 | [-32.768, 32.768] | f(x*)=0, x*=(0, 0) |
| f9 | Rosenbrock | 2 | [-1.2, 1.2] | f(x*)=0, x*=(1, 1) |
| f10 | Schaffer | 2 | [-100, 100] | f(x*)=0, x*=(0, 0) |
| f11 | DeJong 1st | 3 | [-5.12, 5.12] | f(x*)=0, x*=(0, 0, 0) |
| f12 | Rastrigin | 3 | [-5.12, 5.12] | f(x*)=0, x*=(0, 0, 0) |
| f13 | Rosenbrock | 3 | [-2.048, 2.048] | f(x*)=0, x*=(1, 1, 1) |
| f14 | Schwefel | 4 | [-500, 500] | f(x*)=0, x*=(420.9687, 420.9687) |
| f15 | Ackley | 6 | [-32.768, 32.768] | f(x*)=0, x*=(0, …, 0) |
| f16 | Hypersphere | 10 | [-5.12, 5.12] | f(x*)=0, x*=(0, …, 0) |

TABLE. 3 PARAMETERS FOR THE ALGORITHMS

| EBA | | FAS-BA | |
|---|---|---|---|
| *n* | 18 | *nc* | 117 |
| *m* | 10 | *sp* | 15.4% |
| *e* | 3 | *bp* | 55.6% |
| *nre* | 19 | *ep* | 33.3% |
| *nrb* | 6 | *erb* | 3:1 |
| *ngh* | serch rang/2 | *ngh* | search rang/2 |
| *stlim* | 10 | *stlim* | 10 |

### Results and discussions

Table 4 contrasts the results obtained by the proposed FAS-BA with EBA (The basic Bees Algorithm is not included in the experiment because EBA has shown superiority to the basic version in many literatures). The average performance of the algorithms on each benchmark function is reported.

The results reported in boldface stand for the best performance (the least iteration cycles) obtained by the algorithms. To make the comparison more convincing, statistical significance of the difference in iteration cycles between FAS-BA and EBA is evaluated through student's *t*-test. The *t*-tests are run with a confidence level of 95% ($\alpha$=0.05). In Table 4, a "+" symbolizes that the difference of the average iteration cycles between the two versions is statistically significant, whereas a "-" means the difference is not statistically significant. It is observable that FAS-BA outperforms EBA on 14 cases out of 16 in terms of iteration cycles consumed. EBA produces better results only on the functions f2 and f9. However, the statistical test shows that even through EBA requires less iteration cycles to find the optima for the function f2, its performance is not significantly different from FAS-BA.

The functions f4, f5, f6 and f11 are relatively simple problems. It can be observed that FAS2-BA spends the least computation cost on finding the optima. Only on the function f4 it obtains the result that statistically differs from EBA. As to f5, f6 and f11, all the algorithms do not demand too much iteration

cycles to locate the minimum, and hence they exhibit similar performances.

The functions f1, f7 and f8 have shown a little more complexity as the algorithms require approximately 50 iteration cycles to locate a desired solution. The results show that FAS-BA needs less computation cost on f1 and f7, and this improvement is statistically validated by the *t*-test. Individually, 12.23 and 4.55 iteration cycles on average are saved by FAS2-BA, which yields the best performance, while 6.9 and 4.37 less iteration cycles are demanded by FAS1-BA. The algorithms perform similarly on f8 as indicated by the *t*-test.

TABLE. 4 ITERATION CYCLES NEEDED TO OPTIMIZE EACH FUNCTION AND *T*-TEST RESULTS

| No. | EBA mean of itr. cycles | FAS1-BA mean of itr. cycles | λ | *t*-test | FAS2 -BA mean of itr. cycles | λ | *t*-test |
|---|---|---|---|---|---|---|---|
| f1 | 51.35 | 44.45 | 1/4 | + | **39.12** | 1/4 | + |
| f2 | **27.13** | 27.43 | 1/6 | - | 27.51 | 1/5 | - |
| f3 | 298.32 | **240.70** | 1/4 | + | 244.08 | 1/4 | + |
| f4 | 40.76 | 39.76 | 1/4 | - | **39.15** | 1/4 | + |
| f5 | 14.03 | 14.34 | 1/4 | - | **14.00** | 1/3 | - |
| f6 | 19.82 | 19.98 | 1/3 | - | **19.28** | 1/3 | - |
| f7 | 44.75 | 40.38 | 1/5 | + | **40.20** | 1/3 | + |
| f8 | 54.75 | **54.64** | 1/3 | - | 55.2 | 1/3 | - |
| f9 | **16.70** | 18.65 | 1/6 | + | 18.36 | 1/6 | + |
| f10 | 244.02 | **177.04** | 1/3 | + | 184.96 | 1/6 | + |
| f11 | 24.51 | 24.40 | 1/3 | - | **24.24** | 1/3 | - |
| f12 | 272.88 | **193.74** | 1/5 | + | 205.66 | 1/5 | + |
| f13 | 595.82 | **372.06** | 1/3 | + | 378.80 | 1/4 | + |
| f14 | 365.52 | **217.50** | 1/3 | + | 244.90 | 1/3 | + |
| f15 | 264.32 | **211.22** | 1/4 | + | 237.82 | 1/3 | + |
| f16 | 156.00 | 107.24 | 1/6 | + | **96.78** | 1/5 | + |

The functions f3, f10, f12 to f16 are complex functions whose optima cannot be easily located. Nevertheless, the proposed FAS1-BA and FAS2-BA have particularly shown their remarkable advantages on these functions over EBA, as demonstrated in Figure 1. The bar graph looks at the search speeding of the algorithms on the seven complex functions from the benchmark function set. Normally the algorithms require hundreds of iteration cycles to find the individual minimum if they are run with the colony size specified in Table 3 (using a different colony size would lead to a variation in the iteration cycles for finding an optimum). It is observable that the algorithm without FAS needs more iteration cycles (higher bar) to obtain the optimum, and the algorithms based on two FAS procedures do not vary too much. To be more specific in the comparison, the data in Table 4 show that 19.3% and 27.4% computation costs are saved on f3 and f10 respectively by FAS1-BA, while 18.2% and 24.2% are saved by FAS2-BA. The FAS1-BA requires the least iteration cycles on f12 to f15, reducing 29.0%, 37.6%, 40.5% and 20.1% iterations respectively. Correspondingly, the FAS2-BA requires 24.6%, 36.4%, 33.3% and 10.0% less iteration cycles on them. In addition, the FAS2-BA expends the least iterations on finding the optima for the function f16, saving 37.9% computations compared with EBA, and the FAS1-BA needs 31.3% less than this improved version. The improvement brought by both FAS1 and FAS2 on f3, f10, f12 to f16 has been demonstrated as statistically significant.

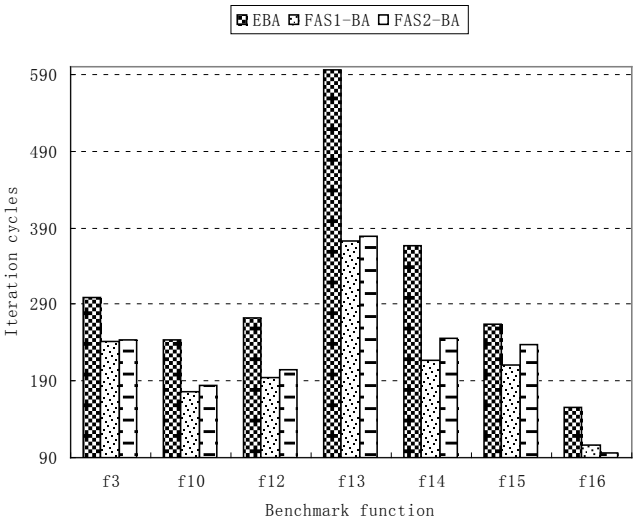


FIGURE.1 COMPARISON OF ITERATION CYCLES REQUIRED TO OPTIMIZE COMPLEX FUNCTIONS

From Table 3 it can be calculated that the colony size of EBA is kept constant as 117, and the function evaluation costs required in every iteration cycle is 99 ($e \bullet nre+(m-e) \bullet nrb$). The initial colony of FAS-BA is set to the same size with EBA. However, the function evaluation costs required in every iteration cycle is variable due to the forager regulation of FAS. The evaluation cost for each function is recorded in the experiment and compared with EBA, as demonstrated in Figure 2.

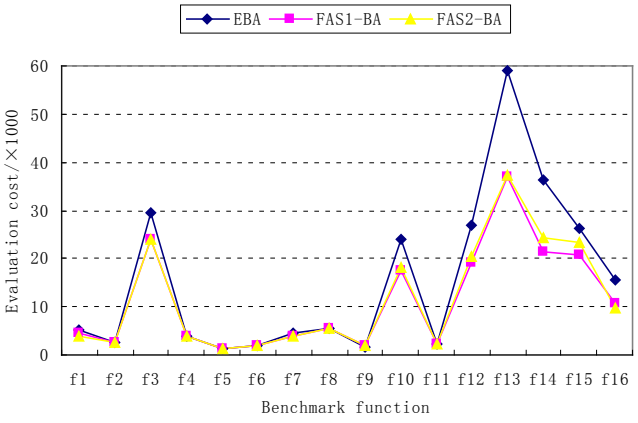


FIGURE.2 COMPARISON OF FUNCTION EVALUATION COSTS REQUIRED TO OPTIMIZE EACH FUNCTIONS

It can be observed in Figure 2 that FAS-BA demands less function evaluation cost for the functions, particularly for the function 3, 10, 12, 13, 14, 15 and 16. FAS1 and FAS2 perform generally the same in saving function evaluation costs. Therefore, the above analyses reveal that the FAS can be

considered as a prominent strategy to enhance the performance of the Bees Algorithm. It can help reduce the expense on computation effectively for solving complex problems, meanwhile, the algorithm using it performs generally no worse than EBA on simple problems.

## A CASE STUDY

The realization of multi-user resource service composition using the proposed intelligent algorithm in cloud manufacturing environment is briefly demonstrated in this section. It is presumed that in a situation three customers post their task requests at a time. The cloud platform is demanded to allocate component services optimally to work out a solution which is up to requirements of as more users as possible. Multi-user RSC appears to be simply repetitive implementation of single user RSC strategies. However, the composite schema should actually take into account much more impact factors. The schema does not necessarily provide each user with the best service chain but strive to satisfy all users.

Other preconditions of this case study encompass: every task requested can be decomposed into 7 subtasks, and there are 15 candidate services available for completing each subtask. It can then be calculated that the size of solution space is $15^7 \times 14^7 \times 13^7 \approx 1.13 \times 10^{24}$. Provided a CPU is able to calculate at the speed of 500 million times per second, it will take more than 7 million years for traditional method to work out the best solution to this NP-hard problem.

For the algorithm, a matrix solution schema is employed, and parameters are empirically set as *nc*=35, *sp*=28%, *bp*=40%, *ep*=25%, *erb*=2, *ngh*=0.6, *stlim*=7, and the fitness factor λ is kept to be $^1/_4$ throughout this study. For simplicity, and to save space, only success rate is reported as shown in Figure 3. The success rate refers to the ratio of users whose requirements have been successfully satisfied against users in total. All statistics are obtained on a repetitive run of the algorithm for 75 times.

Figure 3 shows the advantage of FAS-BA over EBA in terms of success rate when users have different requirements. Although the success rate declines with users' task requirements becoming tougher, the two algorithms using FAS outperform EBA in that they promote the success rate by 11.4% on average. And in terms of time consumed to find an optimal or near optimal solution, FAS-BA costs statistically 45.2 seconds to obtain a solution up to the task in a run, that is 15.8 seconds less than EBA. The two implementations of FAS do not demonstrate distinct difference from each other in this case study, similar to how they perform in numerical experiments.

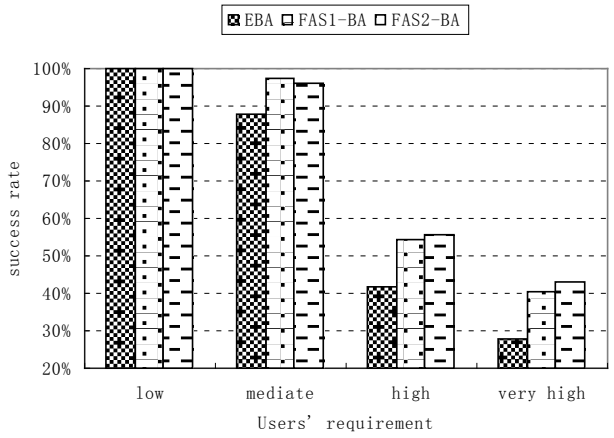


FIGURE.3 COMPARISON OF SUCCESS RATE IN SOLVING RSC PROBLEM

## CONCLUSIONS

The enhancement of intelligent optimization algorithm's performance underpins the promotion of efficiency in solving many problems confronted in cloud manufacturing. The paper has presented a novel strategy called FAS, which takes the inspiration from the natural behaviors of adaptive labor division in honeybee colony, to improve the performance of the Bees Algorithm. Based on the proposed strategy, a new variation of the Bees Algorithm named FAS-BA is developed. The performance of the proposed algorithm is demonstrated on a set of well-known optimization benchmark functions and a case study. The improvement brought by the proposed FAS is highlighted by comparing the experiment results with the Bees Algorithm improved by strategies of neighborhood shrinking and site abandonment. The FAS is shown to be particularly helpful for the algorithm to solve complex problems. To locate the optima of those problems, the FAS-BA requires less computational cost, indicating a promotion of optimal solution search efficiency. The results of *t*-test have proved the promotion is obtained with statistical significance. However, the design of the fitness factor λ may not be the most appropriate way to achieve adaptive forager division in the algorithm. How it affects the performance of the algorithm warrants further investigation. Furthermore, the algorithm demands parameters to be set empirically, how to make the algorithm less reliant on prior knowledge is of great help in current developing wave of science and technology.

## ACKNOWLEDGMENTS

This research is supported by the Keygrant Project of Chinese Ministry of Education (Grant No. 313042), the Key Project of Natural Science Foundation of Hubei Province of China (Grant No. 2013CFA044), the Fundamental Research Funds for the Central Universities (Grant Nos. 2014-VII-015 and 2014-zy-107), and Research Exchange with China and

India, The Royal Academy of Engineering, UK (Grant No. 1415-1)

## REFERENCES

[1] Dorigo, M., Stutzle, T., 2004. “Ant colony optimization”, MIT press, Cambridge, MA.

[2] Tang, K. S., Man, K. F., Kwong, S., He, Q., 1996. “Genetic algorithms and their applications”, *IEEE Signal Processing Magazine*, 13(6), 22-37.

[3] Kennedy, J., Eberhart, R., 1995. “Particle swarm optimization”, *IEEE International Conference on Neural Networks*, 4(1995), 1942-1948, IEEE press, Perth, WA.

[4] Simon, D., 2008. “Biogeography-based optimization”, *IEEE Trans. on Evol. Comput.*, 16(6), pp.702-713.

[5] Karaboga, D., 2005. “An idea based on honey bee swarm for numerical optimization”, Technical report No. TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

[6] Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M., 2006. “The bees algorithm – a novel tool for complex optimization problems”, *Proc. 2nd International Virtual Conference on Intelligent Production Machines and Systems*, Oxford: Elsevier, pp.454-459.

[7] Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M., 2005. „The bees algorithm”, Technical note, Manufacturing Engineering Center, Cardiff University, UK.

[8] Pham, D. T., Castellani, M., 2009. “The bees algorithm: modeling foraging behaviour to solve continuous optimization problems”, *Proc, ImechE. Part C*, 223(12), pp.2919-2938.

[9] Packianather, M. S., Landy, M., Pham, D. T., 2009. “Enhancing the speed of the bees algorithm using pheromone-based recruitment”, *7th IEEE International Conference on Industrial Informatics*, pp.789-794, IEEE, Cardiff, Wales.

[10] Pham, D. T., Darwish, A. H., “Fuzzy selection of local search sites in the bees algorithm”, *4th International Conference on Innovative Production Machines and Systems* [online], http://conference.iproms.org.

[11] Seely, D. T., 1996. “The wisdom of the hive: the social physiology of honey bee colonies”. Harvard University Press, Cambridge, Massachusetts.

[12] Tereshko, V., Loengarov, A., 2005. “Collective decision-making in honey bee foraging dynamics”, *J. Comput. Inf. Syst.* 9(3), pp.1-7.

[13] Yuce, B., Packomamther, M. S., Mastrocinque, E., Pham, D. T., Lambiase, A., 2013. “Honey bees inspired optimization method: the bees algorithm,” *Insects*, 2013, 4(4), pp.646-662.

[14] Castellani, M., Pham, Q. T., Pham, D. T., 2012. “Dynamic optimization by a modified bees algorithm,” *Proc. IMechE. Part I: J. Syst. Control Eng.*, 226(7), pp.956-971.

[15] Yuce, B., Mastrocinque, E., Lambiase, A., Packianather, M. S., Pham, D. T., 2014. “A multi-objective supply chain optimization using enhanced Bees Algorithm with adaptive neighbourhood search and site abandonment strategy,” *Swarm and Evolution Computation*, 18(2014), pp.71-82.

[16] Ang, M. C., Ng, K. W., Pham, D. T., 2013. „“Combining the Bees Algorithm and shape grammar to generate branded product concepts”, *Proceedings of Institution of Mechanical Engineers, Part B*, 2013, 227(2013), pp.1860-1873, ISSN 0954-4054.